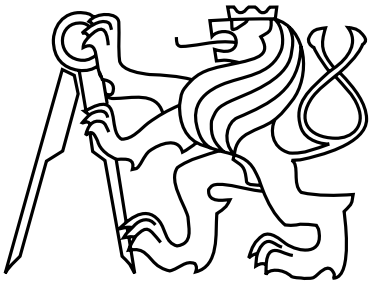
České vysoké učení technické v Praze
Fakulta stavební
Katedra geomatiky



Cvičení 4

Úloha č. 4: Množinové operace s polygony

Bc. Petr Poskočil a Bc. Marek Fáber

Vyučující: doc. Ing. Tomáš Bayer, Ph.D.

Studijní program: Geodézie a kartografie, Navazující magisterský

Obor: Geomatika

3. února 2020

Obsah

Kapitola 1

Zadání

Úloha č. 4: Množinové operace s polygony

Vstup: množina n polygonů $P=\{P_1,...,P_n\}$.

Výstup: množina m polygonů $P'=\{P'_1,...,P'_m\}$.

S využitím algoritmu pro množinové operace s polygony implementujte pro libovolné dva polygony $P_i, P_j \in P$ následující operace:

- Průnik polygonů $P_i \cap P_j$,
- Sjednocení polygonů $P_i \cup P_j$,
- Rozdíl polygonů: $P_i \cap \overline{P}_j$, resp. $P_j \cap \overline{P}_i$.

Jako vstupní data použijte existující kartografická data (např. konvertované shape fily) či syntetická data, která budou načítána z textového souboru ve Vámi zvoleném formátu.

Grafické rozhraní realizujte s využitím frameworku QT.

Při zpracování se snažte postihnout nejčastější singulární případy: společný vrchol, společná část segmentu, společný celý segment či více společných segmentů. Ošetřete situace, kdy výsledkem není 2D entita, ale 0D či 1D entita.

Pro výše uvedené účely je nutné mít řádně odladěny algoritmy z úlohy 1. Postup ošetření těchto případů diskutujte v technické zprávě, zamyslete se nad dalšími singularitami, které mohou nastat.

Hodnocení:

Krok	**Hodnocení**
Množinové operace: průnik, sjednocení, rozdíl	20b
Konstrukce offsetu (bufferu)	*+10b*
Výpočet průsečíků segmentů algoritmem Bentley & Ottman	*+8b*
Řešení pro polygony obsahující holes (otvory)	*+6b*
Max celkem:	**44b**

Obrázek 1.1: Zadání cvičení č. 4 [?]

1.1 Údaje o úlohách

Povinnou částí úlohy je na množině dvou polygonů provést operace průniku, sjednocení a rozdílu. Z bonusové části nebyla zpracována žádná z úloh. [?]

Kapitola 2

Popis problému

Cílem úlohy je vytvořit aplikaci, do které bude možno nakreslit dva polygony A a B. Nad těmito polygony bude možné provádět Boolovské operace - sjednocení, průnik a rozdíl.

- *S*jednocení je operace, kdy výsledkem je množina, která obsahuje všechny části, které jsou obsaženy alespoň v jedné z množin.
- *P*růnik je operace, kdy výsledkem je množina, které obsahuje části, které jsou pro všechny množiny společné.
- *R*ozdíl je operace, kdy výsledkem je množina, která obsahuje části, které jsou součástí pouze jedné množiny.

Kapitola 3

Popis použitých algoritmů

Pro tvorbu Booleovských operací je potřeba nejprve spočítat průsečíky obou polygonů a ohodnotit je podle pozice. Podle ohodnocení se průsečíky spojí do fragmentů, které se nakonec spojí do výsledných polygonů.

3.1 Výpočet průsečíků

Ve výpočtu dochází k určování vzájemného vztahu hran polygonů funkcí get2LinesPosition. Tato funkce určí, zda se hrany protínají, či nikoliv. Pokud se hrany protínají, funkce spočítá jejich průsečík a uloží ho do proměnné datového typu mapa. Do této promněnné se k výsledku ukládá také klíč, který odkazuje k dané hodnotě.

3.1.1 Implementace algoritmu

1. $for(i = 0; i < n; i++)$
2. Vytvoření mapy: $M = map(double, QPointFB)$
3. $for(j = 0; j < m; j++)$
4. Existuje průsečík: $if(b_ij = (p_i, p_(i+)1\%n \cap q_j, q_(j+)1\%m)$
5. Přidej průsečík do M
6. Zpracuj první průsečík pro e_j : $ProcessIntersection(b_ij, \beta, B, j)$
7. Nějaké průsečíky jsme nalezli: $if(||M|| > 0)$
8. Procházej průsečíky v M: $for \forall m \in M$
9. Získej 2. hodnotu z páru: $b \leftarrow m.second$
10. Zpracuj první průsečík pro e_i: $ProcessIntersection(b, \alpha, A, i)$

Process Intersectin

1. $if(|t| < \epsilon)$
2. Startovní bod průsečíkem: $P[i] \leftarrow inters$
3. $if(|t-1| < \epsilon)$
4. Koncový bod průsečíkem: $P[(i+1)\%m] \leftarrow inters$
5. else
6. Inkrementace pozice: $ProcessIntersectini < -i+1$
7. Přidej průsečík na pozici i+1: $ifP \leftarrow (b, i)$

3.2 Fragmenty

Podle ohodnocení hran jsou vytvořeny fragmenty, nebo-li seznamy po sobě jdoucích bodů se stejným ohodnocením. Následně se fragmenty spojí do výsledných objektů.

3.2.1 Implementace algoritmu

1. $i \leftarrow 0$
2. Dokud P[i] není průsečíkem s orientací g: $while(g(P[i]) \neq g \vee P[i] \neq inters)$
3. $i \leftarrow i+1$
4. Neexistuje bod s touto orientací: $if(i \equiv n)return$
5. Zapamatuj index prvního průsečíku: $i_s \leftarrow i$
6. *do*
7. Prázdný fragment $f = \oslash$
8. Nalezen fragment $if(createFragmentFromVertices(i_s, P, g, if))$
9. Swapuj prvky fragmentu: $if(s)f.reverse()$
10. Přidej fragment do mapy: $F[f[0]] \rightarrow f$
11. $i \leftarrow (i+1)\%m$
12. Dokud nedojde k počátečnímu průsečíku: $while(i \equiv i_s)$

3.3 Množinové operace

Postup jednotlivých operací je spojen do výsledného algoritmu.

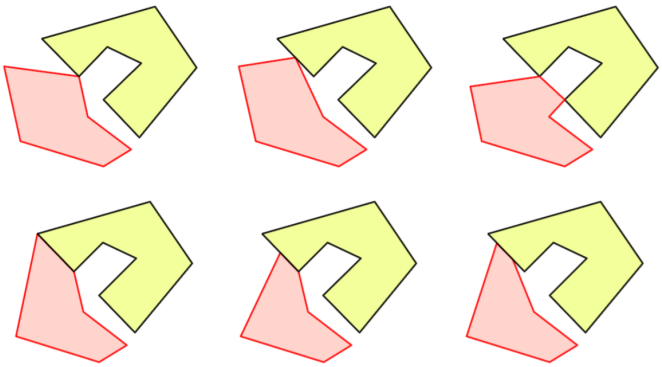
3.3.1 Implementace algoritmu

1. $if(o(A)! = CCW)$
2. Změň orientaci na CCW: $A.switchOr()$
3. $if(o(B)! = CCW)$
4. Změň orientaci na CCW: $B.switchOr()$
5. Urči průsečíky A, B: $ComputeIntersection(A, B)$
6. Urči polohu vrcholů vůči oblastem $SetPositions(A, B)$
7. $map < QPointFB, pair < bool, vector < QPointFB >>> F$
8. $pos1 = (oper \equiv intersection \vee oper \equiv DifAB?Inner : Outer)$
9. $pos2 = (oper \equiv intersection \vee oper \equiv DifAB?Inner : Outer)$
10. $swap1 = (oper \equiv DifAB? : true : false)$
11. $swap2 = (oper \equiv DifAB? : true : false)$
12. $CreateFragments(A, pos1, swap1, F)$
13. $CreateFragments(B, pos2, swap2, F)$
14. $MergeFragments(A, B, C)$

3.4 Problematické situace

Problematická situace může nastat v případě, kdy polygony obsahují nějaké otvory. Tyto otvory by se mohli zařadit do výsledku řešení i přes to, že nejsou součástí vstupního polygonu.

Jelikož je k výpočtu použit Winding Number algorithms, je třeba počítat i se singularitami tohoto algoritmu. Můžou nastat situace, kdy polygony mají společný vrchol, hranu nebo vrchol jednoho polygonu leží na hraně polygonu druhého.



Obrázek 3.1: Singularity [**?**]

Pokud dojde k singulárnímu případu, kdy mají polygony společný bod, tak pro požadavek Intersect je výsledkem prázdná množina hran. V případě, že polygony mají společné hrany, je výsledkem množina těchto hran (viz obr. 4.11). Pro požadavek Union u polygonů se společnou hranou je výsledkem polygon, který společnou hranu neobsahuje (viz obr. 4.10). Pro ostatní množinové operace je z podstaty jednotlivých operací výsledek klasický.

Kapitola 4

Vstup a výstup aplikace

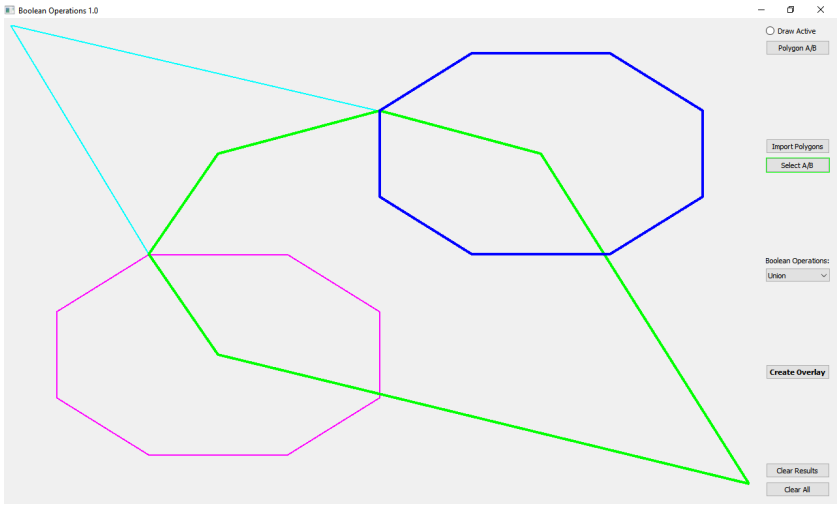
4.1 Vstup

Vstupními daty je množina polygonů, kterou uživatel může ručně nakreslit do grafické části aplikace nebo polygony do aplikace importovat z textového souboru.

4.2 Výstup

Výstupem úlohy je grafická aplikace, která nad vytvořenými polygony provádí množinové operace průniku, sjednocení a rozdílu. Do aplikace je možné importovat soubor s více polygony. Po vybrání dvou polygonů lze nad nimi provádět množinové analýzy.

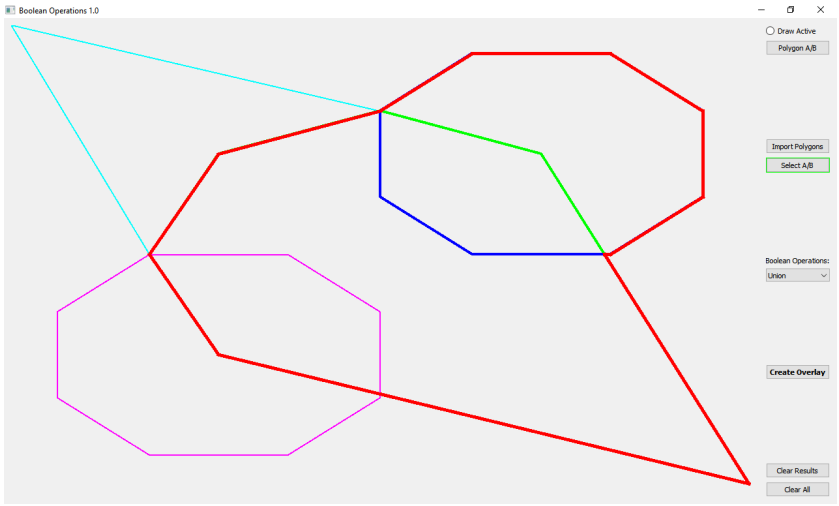
4.2.1 Aplikace



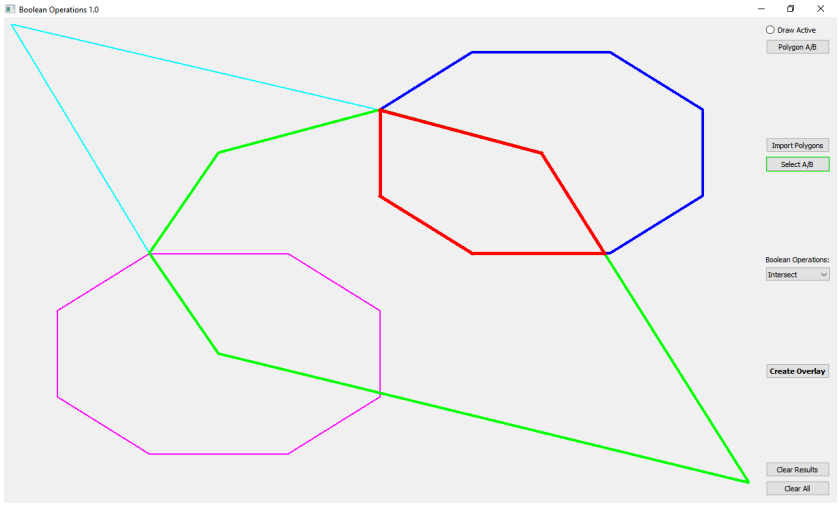
Obrázek 4.1: Ukázka aplikace s naimportovanými polygony

Pro analýzu byl vybrán zelený (A) a modrý (B) polygon. Výsledek je vždy zobrazen červenou barvou.

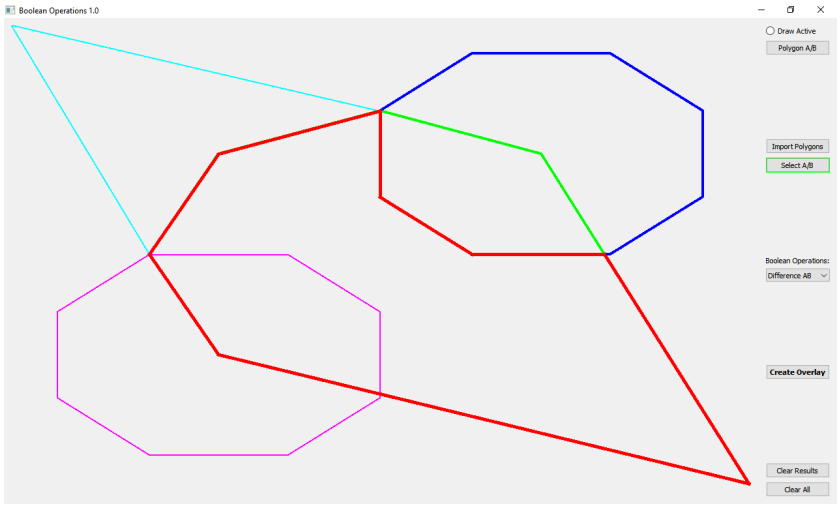
4.2.2 Výsledky analýz



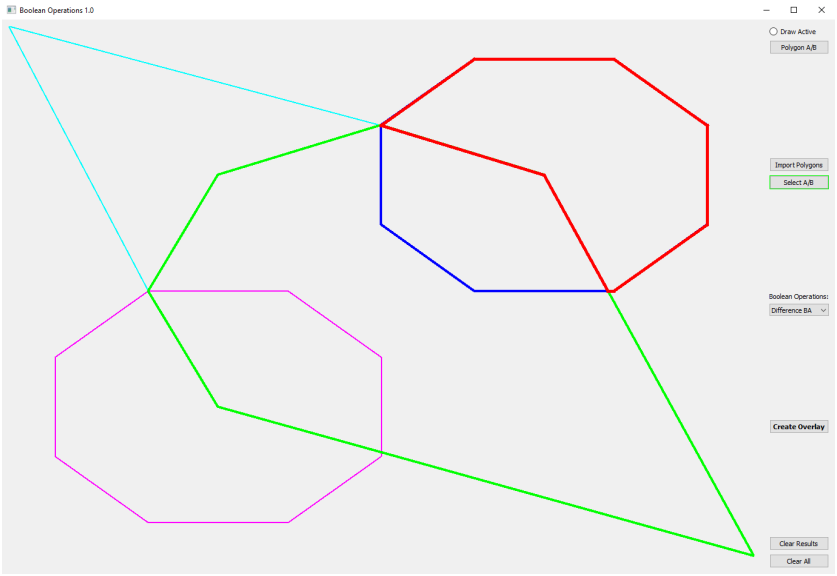
Obrázek 4.2: Sjednocení vybraných polygonů



Obrázek 4.3: Průnik vybraných polygonů



Obrázek 4.4: Rozdíl A-B vybraných polygonů



Obrázek 4.5: Rozdíl B-A vybraných polygonů

4.2.3 Analýza nad ručně nakreslenými polygony

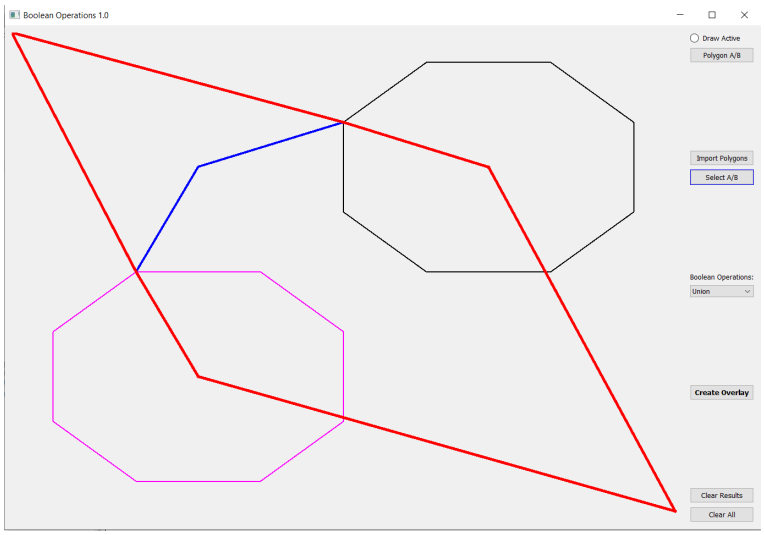
Obrázek 4.6: Sjednocení

Obrázek 4.7: Průnik

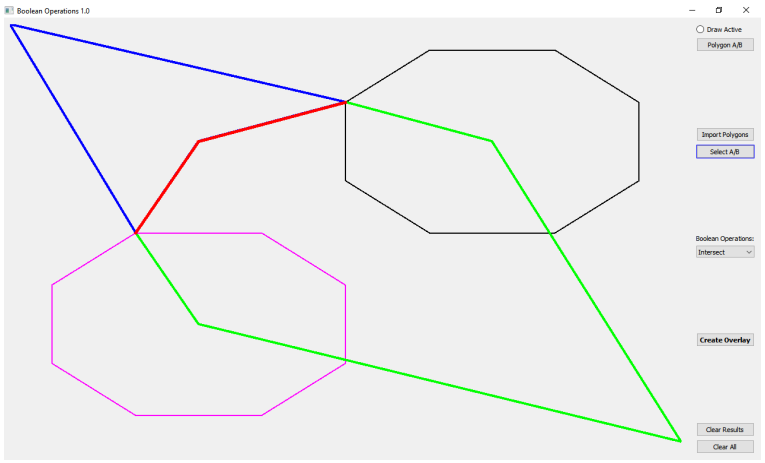
Obrázek 4.8: Rozdíl A-B

Obrázek 4.9: Rozdíl B-A

4.2.4 Singulární případy



Obrázek 4.10: Singulární případ - sjednocení



Obrázek 4.11: Singulární případ - průnik

Kapitola 5

Dokumentace tříd a jejich metod

5.1 Algorithms

- *TPointLinePosition getPointLinePosition(QPointFB &q, QPointFB &p1, QPointFB &p2)*
  Tato funkce určuje pozici bodu vůči linii. Na vstupu funkce je určovaný bod a dva body přímky "a"a "b". Ze dvou bodů přímky můžeme určit determinant, jehož hodnotu porovnáváme se zvolenou minimální hodnotou "eps".

- *double getAngle2Vectors(QPointFB &p1, QPointFB &p2, QPointFB &p3, QPointFB &p4)*
  Tato funkce počítá úhel mezi dvěma hranami. Na vstupu jsou 4 body, které určují dvě přímky. Výstupem je velikost úhlu ve stupních typu double.

- *TPointPolygonPosition positionPointPolygonWinding(QPointFB &q, std::vector<QPointFB> &pol)*
  Tato metoda určí polohu bodu vůči polygonu.

- *T2LinesPosition get2LinesPosition(QPointFB &p1, QPointFB &p2, QPointFB &p3, QPointFB &p4, QPointFB &pi)*
  Metoda, která určí vztah dvou přímek a případně vypočítá jejich průsečík.

- *std::vector<Edge> booleanOperations(std::vector<QPointFB> &polygonA,std::vector<QPointFB> &polygonB, TBooleanOperation operation)*
  Metoda, která provádí Booleovské operace nad polygony.

- *void processIntersection(QPointFB &pi, double &t, std::vector<QPointFB> &polygon, int &i)*
  Metoda vloží bod b do polygonu na pozici i+1, pokud je na hraně daného polygonu a není vrcholem.

- *void computePolygonIntersection(std::vector<QPointFB> &pa, std::vector<QPointFB> &pb)*
  Metoda spočítá průsečíky polygonů a spustí *processIntersection*.

- *int getPositionWindingSelect(QPointF q, QPolygonF polygons)*
  Tato metoda určí polohu bodu vůči polygonu. Slouží k výběru polygonů při nahrání vlastních dat.

5.2 Draw

- *void changePolygon()*
  Metoda, která umožňuje přepínání mezi kresbou polygonu A a B.
- *void setA (std::vector<QPointFB> &a_)*
  Metoda pro nastavení polygonu A.
- *void setB (std::vector<QPointFB> &b_)*
  Metoda pro nastavení polygonu B.
- *std::vector<QPointFB> getA()*
  Metoda vracející polygon A.
- *std::vector<QPointFB> getB()*
  Metoda vracející polygon B.
- *void clearResults()*
  Metoda smaže výsledný polygon Boolovských operací.
- *void clearAll()*
  Metoda, která smaže grafické okno.
- *void mousePressEvent(QMouseEvent *event)*
  Metoda, která slouží ke vkládání bodů polygonu po kliknutí do grafického okna.
- *void paintEvent(QPaintEvent *event)*
  Metoda pro kreslení v grafickém okně.
- *void drawPolygon(QPainter &painter, std::vector<QPointFB> &polygon*
  Metoda, které vykresluje polygony.
- *static void importPolygons(std::string &path, std::vector<QPolygonF> &polygons, QSizeF &canvas_size*
  Metoda, která umožní importovat vlastní polygony.
- *void setRes (std::vector<Edge> res_)*
  Metoda pro nastavení výsledného polygonu Boolovských operací.

5.3 Edge

- *Edge(QPointFB &start, QPointFB &end)*
  Třída odvozená od třídy QPointF, která slouží k uložení bodu s výškou.

- *QPointFB &getStart()*
  Metoda, která vrátí počáteční bod hrany.
- *void setStart(QPointFB &s)*
  Metoda, která nastaví počáteční bod hrany.
- *QPointFB &getEnd()*
  Metoda, která vrátí koncový bod hrany.
- *void setEnd(QPointFB &e)*
  Metoda, která nastaví koncový bod hrany.

5.4 QPointfb

- *double getAlpha ()*
  Vrátí hodnotu alfa bodu třídy QPointFB.
- *double getBeta ()*
  Vrátí hodnotu beta bodu třídy QPointFB.
- *TPointPolygonPosition getPosition ()*
  Vrátí hodnotu position bodu třídy QPointFB.
- *void setAlpha (double alpha_)*
  Nastaví hodnotu alfa bodu třídy QPointFB.
- *void setBeta (double beta_)*
  Nastaví hodnotu beta bodu třídy QPointFB.
- *void setPosition (TPointPolygonPosition position_)*
  Nastaví hodnotu position bodu třídy QPointFB.

5.5 Widget

- *void on_ pushButton_ clicked*
  Umožní přepínat mezi kresbou polygonu A a B.
- *void on_ pushButton_ 2_ clicked*
  Spouští Booleovskou operaci.
- *void on_ pushButton_ 3_ clicked*
  Smaže výsledný polygon v grafické části.
- *void on_ pushButton_ 4_ clicked*
  Smaže polygony v grafické části.
- *void on_ pushButton_ 5_ clicked*
  Umožní načíst vlastní data do aplikace.

- *void on_ pushButton_ 6_ clicked*
  Umožňuje vybrat polygon A a B z vlastních dat.
- *void on_ radioButton_ clicked(bool checked)*
  Aktivuje vlastní kresbu polygonů.

Kapitola 6

Závěr

Ve vytvořené aplikaci lze vytvořit ručně nebo importovat množinu polygonů. Pomocí tlačítka "Select A/B"z importovaný polygonů je nutné vybrat dva polygony, mezi kterými bude docházet k výpočtu.

6.1 Neřešené problémy a náměty

- Z časové tísně nebyly v úloze vyřešeny bonusové úlohy. Zejména by bylo vhodné řešit situaci, kdy polygony obsahují otvory.

Příloha A

Zdrojový kód aplikace

Zdrojový kód aplikace je dostupný z veřejného profilu *petrposkocil* na github.com
Jméno repozitáře:*ADK_poskocil_faber*
Podsložka: *U4*

https://github.com/petrposkocil/ADK_poskocil_faber/U4/BooleanOperations

Obsažené soubory:
BooleanOperations.pro //Qt creator project file
algorithms.h //Header file
draw.h //Header file
edge.h //Header file
qpointfb.h //Header file
types.h //Header file
widget.h //Header file
main.cpp //Source code file
algorithms.cpp //Source code file
draw.cpp //Source code file
edge.cpp //Source code file
qpointfb.cpp //Source code file
widget.cpp //Source code file
widget.ui //User interface file

Příloha B

Aplikace - binární soubor

Aplikace je dostupná z veřejného profilu *petrposkocil* na github.com
Jméno repozitáře: *ADK_ poskocil_ faber*
Soubor: *BooleanOperations.exe*

https://github.com/petrposkocil/ADK_poskocil_faber/U4/BooleanOperations.exe

Příloha C

Výstupní data

Výstupní data jsou ve formě grafického znázornění množinových analýz dvou polygonů.